R. W. Hartenstein, et al.: A Scalable, Parallel, and Reconfigurable Datapath Architecture; Sixth International Symposium on IC Technology, Systems & Applications, ISIC'95, Singapore, Sept. 6-8, 1995

# A SCALABLE, PARALLEL, AND RECONFIGURABLE DATAPATH ARCHITECTURE

Reiner W. Hartenstein, Rainer Kress, Helmut Reinig
University of Kaiserslautern
Erwin-Schrödinger-Straße, D-67663 Kaiserslautern, Germany
Fax: ++49 631 205 2640, email: abakus@informatik.uni-kl.de

## Abstract

*A scalable, parallel, and reconfigurable datapath architecture (rDPA) is presented. Fine grained parallelism is achieved by using simple reconfigurable processing elements which are called datapath units (DPUs). The rDPA is scalable to arbitrarily large arrays and reconfigurable to be adaptable to the computational problem. Pipelining is supported by the architecture. A datapath synthesis system (DPSS) allows automatic mapping of high level descriptions onto the rDPA. The rDPA can be used as a reconfigurable ALU for bus oriented systems as well as for rapid prototyping of high speed datapaths.*

## 1. Introduction

Many computation-intensive algorithms take too much execution time, even on a well-equipped modern workstation. This opens a market for hardware accelerators of all kinds. Custom configurable accelerators have the advantage to be adaptable to the computational problem. Such an accelerator should be scalable. This means that it should be extensible to various sizes depending on the computational needs. Custom computing machines (CCMs) [BuPo94] provide such extensibilities. CCMs are based on SRAM based field-programmable gate arrays (FPGAs) [Fawc95]. Most available FPGAs are configurable only at bit-level to support both random logic as well as datapaths. Consequently they provide only modest performance and capacity with 32-bit datapaths.

The reconfigurable datapath architecture (rDPA) provides higher throughput and more area efficiency for implementation of such wide datapaths than FPGAs available commercially. Therefore it can be used as a basis for building word-oriented CCMs. A controller allows the use of the rDPA as a data-driven reconfigurable ALU (rALU). The rDPA is in-circuit reconfigurable, and it is scalable to nearly arbitrarily large arrays. A synthesis system is available to map statements onto the architecture without user interaction.

## 2. Reconfigurable Datapath Architecture

The reconfigurable datapath architecture (rDPA) consists of reconfigurable processing elements, a sophisticated kind of configurable logic blocks which we call datapath units (DPUs). Connecting an array of two by four rDPA chips on a PCB board, an array of 128 DPUs can be realized (figure 1).The DPUs are interconnected by two interconnection levels. Both, the DPUs and the interconnect are described in the following.

**Datapath unit architecture.** The rDPA consists of a regular array of identical datapath units (DPUs). Each DPU has two input and two output registers. The operation of the DPUs is

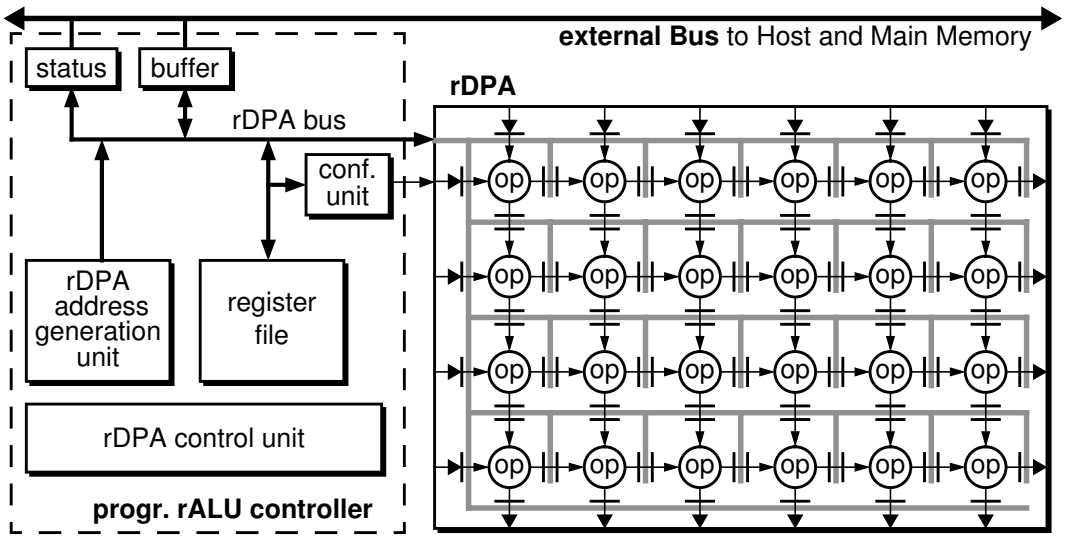


Figure 1. The reconfigurable datapath architecture (rDPA) with the programmable rALU controller

data-driven. This means that the operation will be evaluated as soon as all required operands are available. An extensible repertory of operators for each DPU is provided by the datapath synthesis system form a DPU library. The operator repertory includes the operators of the programming language C. The architecture of the DPUs consists of a datapath including an ALU and a microprogramable control unit. Operators such as addition, subtraction or logical operators can be evaluated directly, and larger operators like multiplication or division are implemented by a microprogram sequence. New operators can be added with the aid of a microassembler.

**Routing architecture.** The rDPA provides two interconnection levels: short lines for local interconnect, and long lines for global interconnect. The topology of an interconnection network can be *static* or *dynamic*. Static networks are fixed during run-time. Dynamic networks can be changed during run-time. Normally in commercial FPGAs all routing resources are static.

The local interconnect of the rDPA is implemented as a mesh. A mesh compared to other array structures is best suited regarding I/O requirements. Further it allows to execute systolic algorithms. These algorithms efficiently use the local interconnect. Although bidirectional communication is more flexible in implementing expressions, a unidirectional approach is used for better area efficiency. A problem occurs with the integration of multiple DPUs onto an integrated circuit because of the high I/O requirements of the processing elements. To reduce the number of input and output pins, a serial link is used for data transfer between neighbouring DPUs on different chips. Internally the full datapath width is used. For the user this serial link is completely transparent.

The global interconnect should provide a connection to each datapath unit (DPU). In the rDPA it is used for I/O of operands from outside into the array, and for propagation of interim results to other DPUs far away. To save area, a time multiplexing of the global interconnect is considered. A scheduling can determine a good usage of this dynamic network. A bus is used for this dynamic interconnect network. A single I/O bus is sufficient to connect all datapath units. Two buses can speed up I/O operations, especially when each DPU has access to both buses, e.g. one input and one output bus. The communication is controlled an external control unit. The data transfers are synchronized data-driven by a handshake like the internal communications.

With the proposed routing architecture, the rDPA can be expanded also across printed circuit board boundaries, e. g. with connectors and flexible cable. Furthermore it is possible to connect the outputs of the east (south) array boundary with the west (north) one, to build a torus. Since the concept is data-driven, the user does not have to worry about synchronization.

**Configuration.** The array is scalable by using several chips of the same type. The DPU address for register addressing of the bus is configured at the beginning. The communication structure allows dynamic in-circuit reconfiguration of the rDPA. This implies partial reconfigurability during run-time. The rDPA can be configured serially from any serial link at the array boundary. One link is sufficient for the complete array, but multiple ports can be used to save time. With the rALU controller also parallel configuration for bus-oriented systems is possible. The configuration is data-driven, and therefore special timing does not have to be considered.

## 3. The rALU Controller

With the rDPA, a programmable rALU controller for bus-oriented systems is provided. Both, the rDPA and the rALU controller form a data-driven reconfigurable ALU (rALU). The rALU controller consists of a rDPA control unit, a register file and an address generation unit for addressing the DPUs (figure 1).

**Register file.** It is useful for optimizing memory cycles, e. g. when one data word of a statement will be used later on in another statement. Then the data word does not have to be read again over the external bus. In addition, the register file makes it possible to use each DPU in the rDPA for operations by using the internal bus for routing.

**Address generation unit.** It delivers the address for the DPU registers before each data is written into the rDPA over the bus. Usually the address is increased by one, but it can also be loaded directly from the rDPA control unit.

**rDPA control unit.** It holds a program to control the different parts of the data-driven rALU. The instruction set consists of instructions for loading data into the rDPA array to a special DPU from the external units, for receiving data from a specific DPU, or branches on a special control signal from the host. The control program is loaded during configuration time. The reconfigurable data-driven ALU allows also pipelined operations.

A status can be reported to the host to inform about overflows or to force the host to deliver data dependent addresses. An input and output buffer decouples the rDPA bus from the external bus.

## 4. The Datapath Synthesis System

The datapath synthesis system (DPSS) allows to map statements from a high level language description onto the rDPA. The statements may contain arithmetic or logic expressions, conditions, and loops, that evaluate iterative computations on a small number of input data.

The task of configuring the rDPA is carried out in the following four phases: logic optimization and technology mapping, placement and routing, I/O scheduling, and finally the code generation. Partitioning of the statements onto the different rDPA chips is not necessary since the array of rDPA chips appears as a single array of DPUs with transparent chip boundaries.

**Logic optimization and technology mapping.** The condition statements are converted into single assignment code. The same is done for the loop conditions. The loop itself is controlled by the rALU controller. Loops and sequences of assignments are considered as basic blocks. Directed acyclic graphs (DAGs) are constructed from the basic blocks. Herewith common subexpressions, identical assignments, local variables, and dead code are removed. Further constant folding and reduction in strength is used. One-input (unary) operators are moved into the next operator if the operator library provides this new merged operator. This step reduces the number of required DPUs in the rDPA array. Further parallelism of single expressions is

increased by tree-height reduction. A simple algorithm is performed which uses the commutativity and the associativity of some operators.

**Placement and routing.** Due to the multiplexed rDPA bus a valid placement is always routable. A poor placement degrades the performance since some internal variables have to be routed via the internal I/O bus. During that time the bus is blocked for other I/O operations. Because of the large number of possible placements, a well developed placement algorithm has to be used. Constructive placement algorithms like the cluster growth algorithm are very fast since no iterations have to be performed. But they usually produce a poorer placement than algorithms that use an iterative improvement method. A well developed iterative improvement algorithm is simulated annealing [Sher93]. The simulated annealing algorithm is easy to handle even if a lot of constraints like in the rDPA have to be considered. It leads to good results in a relatively small amount of time when using it for small problems as the targeted rDPA placement. Further it allows to consider one additional routing operation per local connection in each iteration step. The cost function considers the chip boundaries, the routing operators and with a high cost the connections via the internal I/O bus.

**I/O scheduling.** Due to the global I/O bus of the rDPA array, the loading of the data and the storing are restricted to a single operation per time. An optimal sequence of these I/O operations has to be determined. Furthermore, if the statements of the algorithm belong to an inner loop and are executed several times in direct sequence, pipelining is used to improve speed. On the other side, if the statements belong to an outer loop and are executed once or several times but not in direct sequence, the vectorized expressions are used for different variables and thus improving area. This scheduling is done using a kind of list based scheduling algorithm known from high level synthesis [GDW92].

**Code generation.** The rDPA configuration file is computed from the mapping information of the processing elements and a library with the microprogram code of the operators. The configuration file for the rALU control unit is extracted from the final schedule of the I/O operators.

## 5. Performance

Although the proposed rALU can be used for any bus-oriented host based system, it has been originally designed for the Xputer prototype Map-oriented Machine 3 (MoM-3) [HBK95]. The Xputer system provides a hardware, and a software environment and application development framework for a rALU [HHR91]. The hardware environment consists mainly of a data sequencer [HBK95] which provides a rich repertory of generic address sequences to avoid addressing overhead.

The C compiler for the MoM-3 takes an almost complete subset of ANSI C as input [Schm94]. Only constructs, which would require a dynamic memory management to be run on the MoM-3 have yet not been implemented (pointers, operating system calls and recursive functions). Since the host's operating system takes care of memory management and I/O, the software parts written for execution on the MoM-3 do not need such constructs. No extensions to C language and no compiler directives are required to generate configuration code for the MoM-3. The compiler computes the parameter sets for the data sequencer, and a file for further synthesis towards the configuration code for the rDPA. The compiling and the synthesis works without user interaction.

In the following an example of a two dimensional FIR filter of second order is analysed and the performance is compared to a Sun Sparc 10/51 and an ELTEC host with MC68020 (16 MHz) which serves currently as the host of the MoM-3. The C compiler vectorizes the expression of the FIR filter and performs loop unrolling to the extend of the capacity of the rDPA array. Since the weights are constants, the compiler is able to optimize the multiplications by replacing them with a matching sequence of additions and shift operations.

The two-dimensional FIR filter algorithms are measured for different kernel sizes. Input to the filter is a 1280 by 1024 pixel grayscale image using 8 bits per pixel. The algorithms have been implemented on the different machines, by optimizing the code manually (improving the use of local interconnect and loading of four input words concurrently). Further a larger application example namely the JPEG image compression algorithm is compared. A more detailed description of the MoM-3 implementation of this algorithm can be found in [HBK95]. Table 1 shows the computed performance values and the acceleration factor of the Xputer prototype by using the rDPA as computational element.

| Algorithms (1280 by 1024 pixel image) | 68020, 16 MHz | | Sparc 10, 50 MHz | | MoM-3, 33 MHz |
|---|---|---|---|---|---|
| | time [seconds] | acceleration factor | time [seconds] | acceleration factor | time [seconds] |
| 3x3 2-D FIR filter | 32.11 | 1784 | 0.71 | 39.4 | 0.018 |
| 5x5 2-D FIR filter | 145.90 | 3839 | 1.87 | 49.5 | 0.038 |
| 7x7 2-D FIR filter | 300.40 | 5179 | 3.58 | 62.0 | 0.058 |
| JPEG (704*576 RGB image, 24bits) | 74.50 | 582 | 1.51 | 11.8 | 0.128 |

Table 1. Performance values and acceleration factors of the MoM-3 for FIR filter algorithms with different kernel sizes (in seconds)

## 6. Conclusions

A scalable, parallel and reconfigurable datapath architecture (rDPA) has been presented. The architecture provides higher flexibility, higher throughput and better area efficiency than FPGAs available commercially. The rDPA architecture can be used as reconfigurable ALU for Xputers as well as for rapid prototyping of high speed datapaths. A datapath synthesis system has been implemented which allows the mapping of statements onto the rDPA array without user interaction. For FIR filter and the JPEG compression algorithm, acceleration factors between 12 and 60 have been obtained compared to one of the fastest Sparc stations currently available.

A second realisation of the rDPA using standard cells with a datapath compiler will soon be submitted for fabrication. It provides 32 bit datapaths and arithmetic resources for integer and fixed-point numbers. The datapath synthesis system is completely specified. The optimization, the placement, as well as the scheduling have been implemented. The code generation is currently being implemented.

## References

[BuPo94] D. A. Buell, K. E. Pocek: Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines, Napa, CA, IEEE Computer Society Press, April 1994

[Fawc95] B. K. Fawcett: FPGAs as Configurable Computing Elements; Int. Workshop on Reconfigurable Architectures, Santa Barbara, CA, April 1995

[GDW92] D. D. Gajski, N. D. Dutt, A. C.-H. Wu, S. Y.-L. Lin: High-Level Synthesis, Introduction to Chip and System Design; Kluwer Academic Publishers, Boston, Dordrecht, London, 1992

[HBK95] R. W. Hartenstein, J. Becker, R. Kress, H. Reinig, K. Schmidt: A Reconfigurable Machine for Applications in Image and Video Compression; European Symposium on Advanced Services and Networks / Conference on Compression Techniques and Standards for Image and Video Communications, Amsterdam, March 1995

[HHR91] R. W. Hartenstein, A. G. Hirschbiel, M. Riedmüller, K. Schmidt, M. Weber: A Novel ASIC Design Approach Based on a New Machine Paradigm; IEEE Journal of Solid-State Circuits, Vol. 26, No. 7, July 1991

[Sher93] N. A. Sherwani: Algorithms for Physical Design Automation; Kluwer Academic Publishers, Boston 1993

[Schm94] K. Schmidt: A Program Partitioning, Restructuring, and Mapping Method for Xputers; Ph.D. Thesis, University of Kaiserslautern, 1994